VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

IMPLEMENTACE ON-LINE API PRO ROZPOZNÁVAČ ŘEČI A ANDROID DEMOAPLIKACE

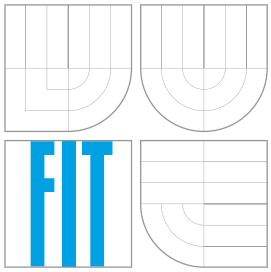
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB GABČO

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

IMPLEMENTACE ON-LINE API PRO ROZPOZNÁVAČ ŘEČI A ANDROID DEMOAPLIKACE

IMPLEMENTATION OF ON-LINE API FOR SPEECH RECOGNITION AND ANDROID DEMOAPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB GABČO
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZŐKE, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

V moderní době se lidé snaží všechno si ulehčit. Tohle může splniť rozpoznávaní řeči. Lokální rozpoznávání řeči je výpočetně náročné, proto se mnoho společností snaží vytvořit vzdálene takzvané síťové rozpoznávaní řeči. V téhle práci se venujeme vytvoření serveru pro rozpoznávání řeči, Android aplikace a výber vhodného protokolu na komunikaci klienta se serverem. Rozebírají se zde prokoly HTTP a Websocketový protokol, rozdíli a výhody mezi nimi.

Abstract

In modern times, people try everything to relieve. This may fulfill speechrecognition. Local speech recognition is computationally demanding, because of that many companies are trying to create a remote network speech recognition. In this thesis, we are focusing on creating a server for speech recognition, Android application and selecting appropriate protocol for communication between client and server. HTTP and Websocket protocol are analyzed her and differences and advantages between them.

Klíčová slova

Android, C++, WebSockety, rozpoznávač hlasu,Phonexia .

Keywords

Android, C++, WebSockets, Speech Recognizer, Phonexia.

Citace

Jakub Gabčo: Implementace on-line API pro rozpoznávač řeči a Android demoaplikace, bakalářská práce, Brno, FIT VUT v Brně, 2015

Implementace on-line API pro rozpoznávač řeči a Android demoaplikace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szőkeho Ph.D.

.....
Jakub Gabčo
20. května 2015

Poděkování

Tímto bych chtěl poděkovat Ing. Igorovi Szőkemu Ph.D. za odbornou pomoc, rady a připomínky při psaní téhle práce.

Obsah

Kapitola 1

Úvod

V moderní technologické době se informační procesy přesouvají z desktopových počítačů a laptopů na menší zařízení jako tablety, inteligentní telefony a další malé doplňky. Všechno plyne do doby, kde se informace předává jenom stlačením tlačítka nebo dotykem na dotykové obrazovce.

Rozpoznávaní řeči a převod textu na řeč se využívá v mnoha informačně-zábavných systémech [1]. Jejich cílem je ulehčit interakci uživatele s těmito systémy. Například: Řidič může zadat mobilní navigaci příkaz. Systém může poskytnou odpověď přes syntézu řeči (napodobování lidské řeči), například převodem textu na řeč. Taktéž, když někdo chce použit mobilní telefon nebo najít cestu, zatímco řídí, může hovořit do svého telefonu a systém najde správnou cestu a tak dále.

Na tyto účely se využívá technologie rozpoznávaní řeči. Tyto systémy jsou plněny gramatikami jazyků. Rozpoznávač hlasu se snaží porovnávat gramatiku se vstupem.

Rozpoznávaní hlasu je náročná záležitost. Existuje mnoho společností jako Google, Nuance, Phonexia[2] a tak dále. Tyto systémy jsou hodně závislé na výpočetní síle a jsou hodně limitovány gramatikou.

Jedním z řešení téhle situace je využívat rozpoznávání řeči vzdáleně, čímž by nebylo potřeba výpočetní síly na slabších zařízeních."Nahraná fráze"je zaslána přes internet na vzdálený server.

Hlavní technologií využívanou v síťové komunikaci s rozpoznávačem je HTTP požadavek / odpověď mechanismus, kde se zvuk posílá po částech v HTTP dotazech v těle zprávy. Rozpoznaný text se vrací v těle zprávy HTTP odpovědi. Další možnou webovou technologií jsou WebSockety.

Cílem této práce je vytvořit efektivní a v reálném čase pracující server pro rozpoznávaní hlasu skrze internet. Implementace je zaměřená na vytvoření rozhraní poskytující klient / server komunikaci a příkladné Android aplikace.

Při vývoji byly využity tyto technologie.

- Phonexia Speech Recognizer
- WebSockety na příjímání/odesílaní
- Android SDK
- C++ Backend Server

[1] www.whyhighend.com/infotainment-system.html
[2] https://www.phonexia.com

Kapitola 2

Rozpoznávaní řeči

Rozpoznávaní řeči, nebo jinak nazývané hlasové rozpoznávaní, je schopnost počítačového programu nebo hardwarového zařízení dekódovat lidský hlas na digitalizovanou zvukovou nahrávku, kterému může porozumět počítač. Rozpoznávače jsou známe pod zkratkami ASR(*Automatic speech recognition*) nebo STT(*Speech to text*). Rozpoznávaní řeči je citlivé na intonaci, výšku hlasu, výslovnosti a jiných faktorech. Rozpoznávače můžeme kvalifikovat jako systémy závislé na mluvčím (SD - *Speaker dependent*) nebo systémy nezávislé na mluvčím (SI - *Speaker Independent*).

Rozpoznávače hlasu, které jsou stavěny na závislosti na mluvčím, potřebují velké množství nahrávek od jednoho mluvčího pro vytvoření dobrých modelů. Tyto systémy dosahují dobré přesnosti rozpoznání.

Rozpoznávače, které jsou nezávislé na mluvčím, jsou natrénovány pomocí nahrávek od velkého množství lidí. Nedosahují ovšem tak dobrých výsledků jako SD systémy. Pro zlepšení výsledků SI systému bylo vyvinuto několik řešení, které adaptují model SI na konkrétního mluvčího [1].

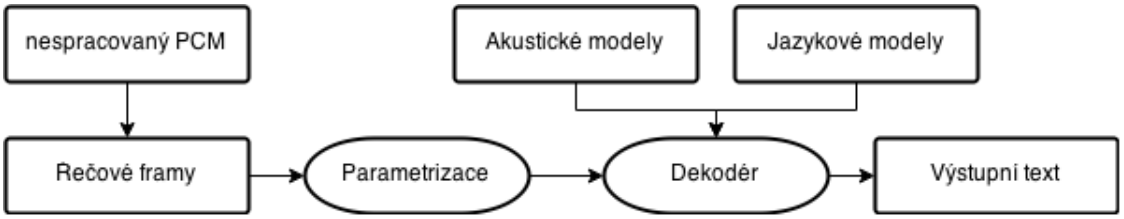
Rozpoznávače hlasu dále můžeme kvalifikovat dle složitosti rozpoznávaní:

- **diskrétní rozpoznávaní řeči** - mluvčí musí čekat mezi jednotlivými slovy, aby systém dokázal rozeznat jednotlivá slova.
- **spojité rozpoznávaní řeči s omezenými slovníky** - využívané při zadávaní stejných informací(čísla, a podobně). Většinou řízeno neuronovou sítí nebo jednoduchými akustickými modely.
- **rozpoznávaní řeči s velkými slovníky** - při rozpoznávaní jsou potřebné informace o akustice a struktuře jazyka(jazykové modely). Využívají fonémy pro rozpoznávaní.
- **rozpoznávaní přirozeného jazyka** - systémy, které nejen dokážou rozeznat řeč, ale dokážou rozumět i otázkám a žádostem, které jím řečník zadal.

Kdysi se používala metoda dynamického borcení času [5] (DTW - Dynamic Time Warping). Tato metoda byla využívaná převážně v rozpoznávačích s diskrétním systémem. Je potřebné mít uloženou každou nahrávku použitého slova. Tato metoda se snaží najít shodu mezi dvěma nahrávkami slov, popřípadě slovních spojení. Při porovnávání se musí brát v potaz rychlost a délka nahrávek. Pro nejlepší výsledky se musí tyto hodnoty vyrovnávat.

V systémech rozpoznávacích řeč se ukázalo jako dobré využití takzvaných Neuronových sítí[3]. Využívají se hlavně pro rozpoznávaní.

[1] http://www.kky.zcu.cz/en/publications/ZbynekZajic_2008_Automatickaadaptace



Obrázek 2.1: Typická struktura rozpoznávače hlasu

V dnešní době, se v moderních systémech, které rozpoznávají řeč, využívají takzvané skryté Markovského modely [1](HMM - *Hidden Markov Models*). HMM je statický Markův model, který modeluje systém za předpokladu, že jde o Markův proces se skrytými stavy. Používají se v systémech rozpoznávaní řeči, protože signál řeči muže být po krátkých časových úsecích chápan jako stacionární signál nebo po částech stacionární signál. To znamená, že na krátkých úsecích může být řeč aproximována jako stacionární signál(10 ms části). Rozpoznávače hlasu využívají tuhle metodu na rozpoznávaní spojitých projevů řeči, čímž stoupá komplexnost a náročnost úloh, než při diskrétním rozpoznávaní řeči. Při mluvení vznikají situace jako zpodobňování hlásek, polykání hlásek nebo náhlá změna intonace a podobně. HMM mohou využívat neuronové sítě na předzpracovaní signálu řeči.

2.1 Zpracování řeči

Zvuk se dá definovat jako mechanické vlnění v látkovém prostředí, které je schopno vyvolat sluchový vjem. Pro lidi se část zvuku projeví jako slyšitelný zvuk, který má frekvenci od 16 Hz do 20 kHz (záleží od konkrétního jedince, jen málokdo je schopen vnímat celé pásmo). Frekvence zvuku nižší než 16 Hz se nazývá infrazvuk a slyší jej například sloni. Frekvence zvuku vyšší než 20 kHz vnímají delfíni, psi či netopýři. Děje, které jsou spojeny se vznikem zvuku, jeho šířením a vnímáním, se nazývají akustika.

2.2 Struktura rozpoznávače

Na obrázku 2.1 jde vidět strukturu běžného rozpoznávače. Skládá se z těchto prvků:

- **Vstup** - na vstupu každého rozpoznávače je v mnoha případech jiný zvukový formát. Proto se obyčejně před vstup vkládá zvukový enkodér. V našem rozpoznávači budeme na vstup posílat raw PCM rámce.
- **"Řečové"rámce** - nad parametry může být provedena první nebo druhá derivace, čímž poskytnou další explicitní informace o dynamice řeči, čímž se může zlepšit výkonnost rozpoznávaní.
- **Parametrizace** - vstupní promluva by měla být transformována a komprimována, aby se zjednodušily následující procesy. Mnoho technik dokáže extrahovat užitečná rysy a data zkomprimovat až s faktorem 10 bez ztráty důležitých informací. Jsou využívané techniky jako FFT,LPCC,PLP,MFCC a tak dále.
- **Akustická analýza a dekódování** - aby se mohly analyzovat rámce řeči, je třeba mít akustické modely. Je mnoho druhů akustických modelů, které se liší v závislosti na své reprezentaci, zrnitosti, kontextu a tak dále. Mezi 2 populární reprezentace akustických

modelů patří předloha(template) nebo stavy(states). Pro co nejlepší rozpoznávání je třeba mít mnoho natrénovaných akustických modelů. U dekódování se využívají akustické modely, jazykové modely a výslovností modely.

- **Výstup** - je rozpoznaná fráze, případně i časová značka fráze.

2.3 Využití rozpoznávačů

Existuje mnoho druhů rozpoznávačů, placených nebo volně dostupných. Každý využívá rozdílný vstupní zvukový formát. Řešení téhle situace je vložit před rozpoznávač hlasu zvukový dekodér/enkodér, který zabezpečí validní vstupní zvukový formát.

Rozpoznávače hlasu využívají gramatické nebo jazykové modely. Gramatika je slovní vzor nebo krátká věta, sestavená podle určitých pravidel. Ta je dodávána rozpoznávači, který na tomto základě určuje význam mluvených slov. Obyčejně je uživatelům dovoleno vložit vlastní vstupní gramatiku s několika pravidly.

Rozpoznávaní řeči není jenom o přepisu mluveného slova. Může být rozšířeno podle potřeb. Některé důležité aplikace rozpoznávaní jazyka:

- Automatický překlad fráze
- Programování jen za pomocí hlasu
- Zautomatizování domácnosti
- Diktování zdravotních záznamů

2.4 Google Speech API

Google Speech API je rozhraní pro využití systému rozpoznávaní od firmy Google. Tento systém je mocný, není však lehce dostupný pro veřejné použití. Google ujasnil, že tato API není veřejně využitelná, a každý kdo bude chtít použít toto API, bude muset dostat souhlas od společnosti Google. Aktuální verze API primárně využívá *Speech Input Javascript API* od Google Chrome, který je definován v *W3C Web Speech Api Specification*[2].

2.4.1 Získání API klíče

Aby jste mohli získat přístup k API klíči pro Google Speech, musíte být členem skupiny Chromium Dev Group[3]. Když budete členem této skupiny, nová služba se vám objeví v Google Cloud Console. Tento API klíč vám dovolí jenom 50 požadavků denně. Je to docela velká limitace. Google Speech API využívá HTTP protokol[4].

2.4.2 Google Chrome API klíč

Jednou z možností je využít API klíč,který se využívá při ukázkové aplikaci[5] a je zabudovaný do každého prohlížeče Google Chrome. Pro zjištění API klíče je potřeba zachytit

[2]https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html, Glen Shires & Hans Wennborg, Google Inc., 19 October 2012, W3C

[3]https://groups.google.com/a/chromium.org/forum/#!forum/chromium-dev

[4]http://geekexplains.blogspot.sk/2008/06/whats-http-explain-http-request-and.html

[5]https://www.google.com/intl/en/chrome/demos/speech.html

HTTP hlavičku, která obsahuje daný klíč. Avšak Chrome komunikuje se Speech API skrze šifrovanou komunikaci TLS/SSL a tím pádem programy jako tcpdump nebo wireshark nebudou užitečné. Proto je vhodné pro tyto účely použít v Chrome zabudovaný nástroj, do kterého se dá dostat napsáním *"about:net-internals"* do řádku s url adresou.

Po zachycení HTTP hlavičky jde zjistit API klíč zabudovaný v Chrome. Zde je možno vidět HTTP hlavičku zasílanou na server:

```
host":"www.google.com",":method":"POST",":path":"/speech-api/full-duplex/v1/
    up?key=AIzaSyBOti4mM-6x9WDnZIjIeyEU21OpBXqWBgw&pair=3B57192471F0F183&
    output=pb&lang=en-US&pFilter=2&maxAlternatives=1&client=chromium&
    continuous&interim
```

Zde je možno vidět nejen API klíč, ale i cestu a parametry, které Chrome využívá při HTTP požadavku. API klíč je "zabudován"v prohlížeči Chrome. Jednou z dobrých věcí na tomto API klíči je, že nemá limit na počet požadavků za den, protože tento klíč "využívají"všichni uživatelé prohlížeče Chrome.

2.4.3 Použití API

Jednou z prvních informací, které byly veřejné, byla koncová adresa API[6]. Tento koncový uzel povoluje maximálně 40 sekundové nahrávky s limitovanou velikostí souborů. Avšak přesné specifikace nejsou známé. V jednoduchosti je to jenom HTTP POST požadavek na jeh zmíněnou url, kde jsou data přidaná do těla požadavku. Při požadavku je nutno nastavit *Content-Type* v následující hlavičce:

Content-Type: audio/x-flac; rate=16000

Je možné zadat 2 druhy zvuku a to: *FLAC* a nebo *SPEEX*. Je doporučeno použít *FLAC*, protože *SPEEX* nepodporuje standardní formátování. Při volání API je třeba předat ještě další potřebné parametry:

.../speech-api/v1/recognize?xjerr=1&client=chromium&lang=en-US

Jak jde vidět, paramatr lang říká serveru, že jazyk je English-US, tolerance chyb je nastavena na 1 a klient je Chromium. Úspěšný požadavek vrací výsledek ve tvaru JSON, ve kterém je pole možných odpovědí.

```
{
  "status": 0,
  "id": "b3447b5d98c5653e0067f35b32c0a8ca-1",
  "hypotheses": [
  {
    "utterance": "i like unicorns",
    "confidence": 0.9012539
  },
  {
    "utterance": "i like unicorn"
  }]
}
```

[6] https://www.google.com/speech-api/v1/recognize

2.4.4 Obousměrné API

Jednosměrná verze API je limitována na krátké a malé soubory. Přestože je lehká na použití a nepotřebuje API klíč, její funkcionalita je značně omezena. Další druh API je takzvaný **FULL-DUPLEX** API, který využívá i Chrome.

Důležitou věcí, kterou je nutno si pamatovat, je to, že koncový uzel je obousměrný(full-duplex). Proto je nutno mít napojeny 2 požadavky(upstream a downstream), kdy jeden bude zasílat data na server a druhý bude stahovat přepis.

Namísto toho, aby jsme zaslali jeden HTTP POST požadavek s daty a přepis dostali zpět jako odpověď, je potřeba poslat zároveň POST a GET požadavek. POST požadavek je následující:

http://www.google.com/speech-api/full-duplex/v1/up

Je potřeba předat ještě další parametry:

- **key** = API klíč
- **pair** = náhodný řetězec pro spárování POST a GET požadavků
- **output** = typ odpovědi(jestli nezadán, použije se JSON)

První požadavek musí být vždy POST, protože jestli GET nenajde vhodný pár, skončí to chybou. Pro požadavek GET je url následující:

http://www.google.com/speech-api/full-duplex/v1/down

Jedinými potřebnými parametry jsou API klíč a pair, zadaný při POST požadavku.

2.4.5 Zhrnutí

Jak je vidět, Google Speech API je mocný nástroj, který však není dostupný pro veřejnost a můžeme jenom doufat, že ho Google v brzké době vypustí do světa.

2.5 Další rozpoznávače hlasu

Na světe existuje mnoho firem, které vyvíjejí systémy pro rozpoznání řeči. Jedny z dobře kalibrovaných ASR, volně šiřitelných nebo komerčních, jsou Phonexia, Nuance, iSpeech, Lumenvox, atd.

Cílem téhle práce není napsat vlastní rozpoznávač hlasu. Popis implementace bude popsán v kapitole 6.

Kapitola 3

Brno Speech API

Knihovna BSAPI(Brno Speech API) dovoluje používat program Phonexia Speech Transcription[1], který je systémem rozpoznávaní od firmy Phonexia, se sídlem v Brně. Budeme ho používat v naší práci.

3.1 Použití

Jde použít ve vícero oblastech. Mezi hlavní patří:

- Získávání dat se zvukovým záznamem a indexováním pro pozdější vyhledávání
- Přepis konferenčních hovorů pro různé účely jako archivaci, indexování a vyhledávaní, atd.
- Vyhledávaní určité informace ve velkém množství zvukových nahrávek

3.2 Dostupné jazyky

Je možné rozpoznávat v následujících jazycích:

- Angličtina
- Čeština
- Ruština

Phonexia nabízí vývoj přepisu spontánní řeči na text, pro nový jazyk (na 100 hod. audia a jeho přepisu, vývoj 4-6 měsíců).

3.3 Technologie

Je zde implementována poslední generace technik pro akustické modelování, neuronových sítí a techniky automatické adaptace na mluvčího a je založena na poslední generaci rozpoznávačů řeči. Dále je taky kompatibilní s různými druhy audia(aplikuje nejnovější techniky kompenzace kanálů):GSM, 3G, VoIP, satelitní, TV a tak dále. Jazykové modely byly trénované na obrovských textových kolekcích, se zaměřením na mluvení frází.

[1] https://www.phonexia.com/technologies/stt

3.4 Možnosti integrace

Rozpoznávač řeči od společnosti Phonexia dovoluje využívat tento systém při implementaci jiných systému pomocí SDK(software development kit). Dovoluje integrovat v jazycích: API for C++, C# nebo Java; instrukce pro .Net, Delphi, Python a tak dále. Dále je možnost využít verzi pro příkazový řádek nebo *Speech Intelligence Platform* a nebo použít aplikaci s GUI, vhodnou pro testování. SDK je dostupné pro platformy Linux a Windows, jak 32 bitové verze tak i 64 bit verze těchto systémů.

3.5 Vstup a výstup systému

Vstupní formát souborů může být ve tvaru MS Wave, Raw s lineárním kódováním(8 nebo 16 bitů), A-law nebude Mu-law. Primárně nastavená vzorkovací frekvence je 8 kHz. Na vstup může být zadán adresář s audio soubory nebo seznam souborů.

Výstupem rozpoznávaní může být:

- **Nejlepší přepis** - soubor nejlepších variant přepisu s časovými značkami(začátek, konec, pravděpodobnost slova)
- **Varianty přepisu** - soubor s několika variantami přepisu pro každý moment(hypotézy možných slov) s časovými značkami. Je možné dostat výstup v různých formátech.(JSON, a tak dále). Je možné dostat dodatečné informace jako: skóre, délka řeči, soubor, jazyk, a tak dále. Při přepisu se vytváří log se zpracovanými informacemi.

Kapitola 4

Protokoly vhodné na komunikaci

V téhle kapitole se dočtete, jaké protokoly se využívají při síťovém rozpoznávaní řeči, proč jsou některé protokoly vhodnější než jiné a rozdíly mezi nimi. Je potřebné si předem určit výhody a nevýhody daných protokolů, čímž se zlepší možnost výběru.

V informatice můžeme definovat protokoly jak konvenci nebo standard, podle kterého probíhá elektronická komunikace a přenos dat mezi dvěma koncovými body. Mohou být realizované hardwarově, softwarově nebo kombinací obou. Protokoly definují pravidla řídící syntaxi, sémantiku a synchronizaci vzájemné komunikace.

Pod skupinou protokolů jsou síťové protokoly. Internetové protokoly jsou zastřešeny sdružením Internet Engineering Task Force(IETF). Za prudkým rozvojem internetu v posledním desetiletí může taktéž zveřejňování internetových protokolů online, jako veřejně přístupné RFC dokumenty.

4.1 TCP a UDP protokoly

Sítě poskytují 2 druhy služeb: orientované na připojení a nespojované(nevytvářejí spojení). První z nich využívá TCP protokol[2]. Před odesláním reálných dat, si odesílatel a příjemce vymění kontrolní pakety na připravení spojení. TCP poskytuje spolehlivou službu nad IP. To znamená, že pakety jsou přijímány v pořadí, v jakém byly odeslány a bez chyb. Na zajištění tohoto stavu, se po úspěšném přijetí paketu, příjemcem odešle potvrzení. V případě, že odesílatel nepřijme potvrzení o doručení paketu, odešle paket znovu. TCP taktéž zajišťuje kontrolní součet celého TCP protokolu datových jednotek (PDU - *Protocol data unit*).

Nespojované služby zajišťuje UDP protokol[6]. V tomto případe se zde neposílá žádný kontrolní paket jako u TCP spojení. Pakety se zasílají ihned jak jsou k dispozici. Takže, UDP poskytuje jednoduchou službu s dvěmi funkcemi: demultiplexing a zjišťovaní chybovosti. Demultiplexing zajišťuje a dovoluje zaslání paketu pro konkrétní odpovídající program. Aby toto mohlo být zajištěno, UDP hlavička obsahuje port specifické běžící aplikace na cílovém počítači a původní port je přidělen aplikaci, v zdrojové konzoli, která přijímá odpovědi.

Aplikace, které potřebují data v reálném čase obvykle využívají UDP namísto TCP, ačkoli TCP zajišťuje doručení vyslaných paketů, ale nezajišťuje přijmutí dat v přiměřené době. Z tohohle pohledu se pro aplikace, využívající reálná data, zdá lepší použití UDP, protože se zpožděnými nebo ztracenými pakety, se může zacházet tak, aby se negoval jejich negativní účinek.

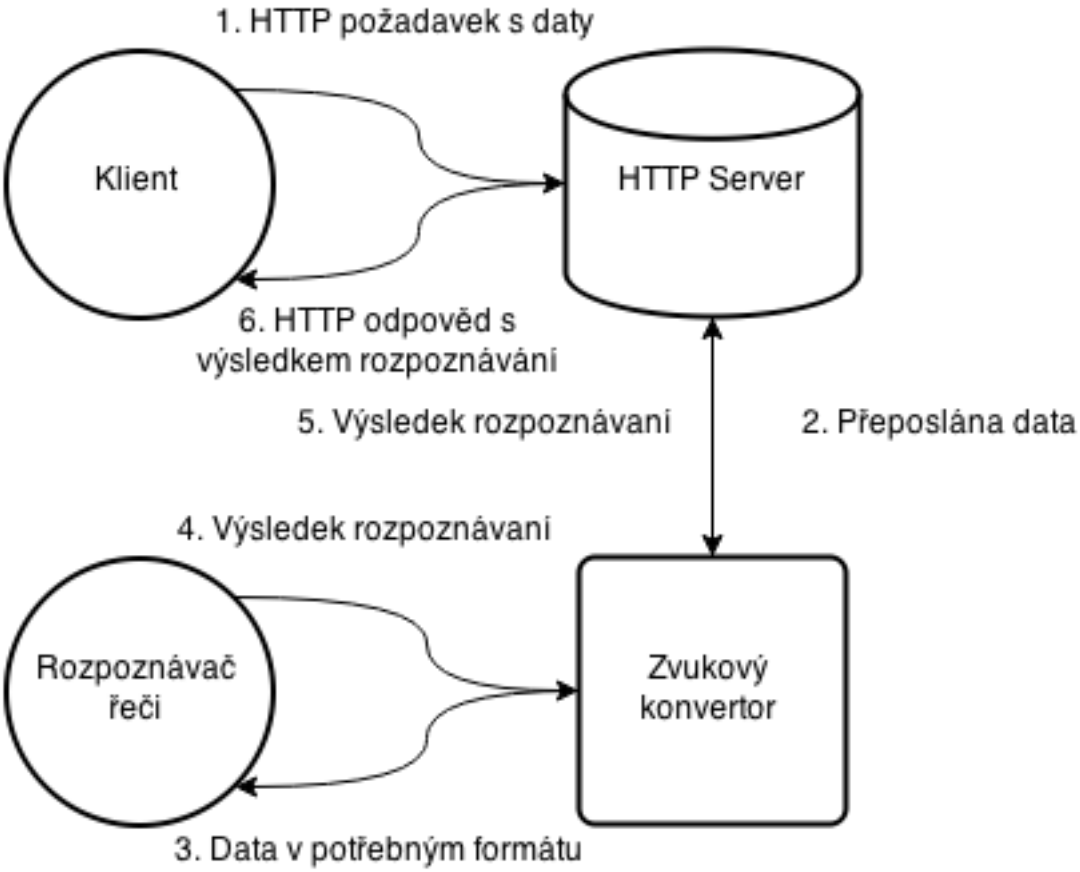
Pro naše řešení není primární cíl pracovat s reálnými daty. Protože u TCP spojení je

zaručeno doručení všech paketů, naše aplikace nebude muset řešit nahrazovaní ztracených paketů, čímž se zlepší přesnost rozpoznávaní na úroveň porovnatelnou s offline řešením.

4.2 Využívané protokoly

Existuje mnoho řešení, které mohou využívat různé protokoly na komunikaci klientu se serverem. Valná většina systémů vzdáleného rozpoznávaní řeči využívá HTTP protokol spolu s RESTfull API. Doba jde dopředu a nově vytvořené webové rozhraní rozpoznávačů řeči využívá Websocketový protokol. Převážně se při použití HTTP protokolu data pravidelně zasílají na server a odpověď je zaslána po skončení nahrávaní hlasu.

4.3 Rozpoznávače založené na HTTP protokolu



Obrázek 4.1: Síťové rozpoznávaní řeči s použitím HTTP dotazu/odpovědi

Obrázek 4.1 zobrazuje použití HTTP dotazu/odpovědi. Síťové rozpoznávaní řeči umožňuje získávat efektivnější a hlavně přesnější výsledky. Celý systém obsahuje rozpoznávací engine(systém), který je spuštěn na výkonném počítači, který má k dispozici mnohem větší gramatiky, které umožňují lepší výsledky než mobilní nebo lokální rozpoznávače hlasu, které mají limitovanou gramatiku a tím pádem méně přesné výsledky.

Síťové rozpoznávače mohou pracovat po LAN, WAN/MAN prostředí. Mohou obsluhovat paralelně velké množství klientů a stále si zachovávat robustní výstup. Valná většina společností využívá tuto architekturu pro své síťové rozpoznávače hlasu. Před detailním vysvětlením, jak tyto systémy fungují, je třeba ještě vysvětlit další věci. Klient může být implementován v jazycích jako Java, Ruby, Python, Perl, C/C++, atd. Klient může taktéž nahrávat zvuk pomocí mikrofonu nebo použit zvukovou nahrávku. Na odesláni nahrávky a přijetí výsledků klient používá HTTP dotaz/odpověď mechanismus.

Při použití HTTP protokolu jsou zasílány na server specifické HTTP hlavičky[1] a tělo zprávy.

Pro rozpoznávání jsou důležité tyto entity s HTTP hlavičky

- Content-Type: audio/x-wav;codec=pcm;bit=18;rate=16000
- Content-Language: cs

Server přijme HTTP dotaz a po zpracováni pošle výsledek rozpoznávaní jako HTTP odpověď. Na obrázku 4.1 je nahrávka od klienta přeposlána do zvukového konvertoru, který zvuk upraví na potřebný formát, požadovaný rozpoznávacím systémem. Ten je hlavní částí celého síťového rozpoznávaní řeči.

Celé schéma začíná u klienta. Ten si pomocí mikrofonu nahraje zvukovou nahrávku nebo použije uložený soubor. Vytvoří HTTP dotaz spolu s nahrávkou jako tělem zprávy a odešle ho na server. Ten přijme HTTP dotaz a přepošle ho konvertoru na zpracování. Konvertovanou zvukovou nahrávku přepošle do rozpoznávače řeči, který vrátí výsledek rozpoznávání a přepošle ho na server.Ten pomocí HTTP odpovědi s výsledkem rozpoznávaní, jako tělem zprávy, pošle výsledek klientovi. Klient přijme výsledek rozpoznávaní.

4.4 WebSockety

Sockety jsou způsobem, jak komunikovat mezi dvěmi aplikacemi, nebo různými zařízeními. V UNIXovém klient/server programování sockety využívající TCP spojení, umožňují obousměrnou komunikaci po jednom TCP spojení tzv. Full Duplex[7].

Dlouhou dobu nebyl žádný koncept socketů nad webovými technologiemi, dokud nebyly vytvořeny websockety. Websockety[2] mohou být použity ve dvou internetových prohlížečích nebo ve webové aplikaci, nasazené na webserveru, Užíváním oboustranné komunikace skrze jedno TCP spojení, vytváří přenos dat v reálném čase. Websocket protokol je nezávislý, na TCP založený, protokol. Data přenášená mezi klientem a serverem se dají posílat jako UTF-8 text, binární rámce nebo speciální kontrolní rámce, používané na obsluhu spojení. Při navazovaní spojení tzv. handshake se postupuje následovně:

- WebSocket pošle handshake žádost na server
- Server interpretuje handshake požadavek jako vylepšený požadavek
- Server pošle klientovi vylepšený požadavek

[1] http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
[2] http://www.zdrojak.cz/clanky/web-sockets/

Pro inicializaci websocketového spojení je zapotřebí adresa vzdáleného serveru. URI je identifikované začátkem ws://(wss:// pro zabezpečené spojení). Dále následuje host a číslo portu. Příkladná URI adresa může být ws://host.cz:port.

Předpokládejme, že máme požadavek URL ws://nejaky.host:8000 a naše požadující metoda je GET, tak pak handshake je následující:

WebSoket Handshake požadavek
Get ws://nejaky.host:8000 HTTP/1.1
Connection: Upgrade
Host:localhost:8080
Origin:http://localhost:8080
Sec-WebSocket-Key1: xxx xxx xxx
Sec-WebSocket-Key2:xxx xxx xxx
Upgrade:WebSocket (Key3): XX:XX:XX:XX:XX:XX:XX:XX

WebSoket Handshake odpověd od serveru
Connection: Upgrade
Sec-WebSocket-Location: ws://somehost.com:8000/
Sec-Websocket-Origin: http://localhost:8080
Upgrade:WebSocket
(Challenge Response): XXXXX

Websocket protokol byl standardizovaný IETF(Internet Engineering Task Force) jako RFC 6455 a jeho API bylo standardizované W3C(World Wide Web Consortium). Mnoho prohlížečů podporuje technologii websocketů. Na připojení k serveru stačí využít Javascript.

Data můžou být zaslána na server jako text nebo binární data ve formě BLOBů[3].

```
<script type="text/JavaScript">
  var wsURL = ws://ipadresaservera:8000;
  var wsocket = new WebSocket(wsURL);
  // Při připojení
  wsocket.onopen = function(evt) {
    //Posílaní dat na server
    wsocket.send("Můžme poslat text nebo binární data");
  }
  // Při příchodě textových/binárních dat
  wsocket.onmessage = function(evt) {
  }
  // Při chybě je zavolána tahle funkce
  wsocket.onerror = function(evt) {
  }
  // Při ukončení spojení je zavolána tahle funkce
  wsocket.onclose = function(evt) {
  }
</script>
```

[3]https://developer.mozilla.org/en-US/docs/Web/API/Blob

4.5 Výhoda WebSocketů

V této kapitole jsme se zabývali protokoly, které jsou využívány v síťovém rozpoznávaní řeči. Dozvěděli jsme se, jak celý systém funguje na základě HTTP požadavku/odpovědi a Websocketů.

Při výběru protokolu jsme se dostali na rozcestí a museli jsme vybírat mezi existujícími alternativami.

V následující tabulce můžete vidět rozdíly mezi jednotlivými alternativami těchto systémů.

	Websocketový základ systému	*HTTP dotaz/odpověď základ systému*
Handshake	Používá na standardní Websoketové spojení tzv. "Upgrade Request"v HTTP hlavičce v době handshaku	Používá standardní hlavičky HTTP požadavku
Obsah v reálnem čase	Umožňuje posílaní obsahu na server v reálném čase	Neposkytuje tuto možnost
Princip komunikace	Poskytuje Full-duplex připojení	Neposkytuje Full-duplex připojení
Nízká režie	Data můžou být zaslána přes normální sockety tj. bez nadbytečné hlavičky	Potřebná hlavička HTTP
Posílání zvuku na server	Po vytvoření websocketu, zvuk je zaslán na server, přes něj jako skrze normální soket	1. Vytvoří se HTTP klient 2. Vytvoří se požadavek HTTP 3. Do požadavku se připojí hlavička 4. Do těla zprávy se připojí zvuk
Přijímání zvuku na straně serveru	Zvuk se přijme jako proud bajtů tzv. byte stream	1. Server přijme HTTP požadavek od klienta 2. Zvuk je uložený jako proud bajtů z těla HTTP požadavku
Přeposlání zvuku	Zvuk je zaslán do rozpoznávače	Zvuk je zaslán do rozpoznávače
Rozpoznávání hlasu	Rozpoznávač vrací řetězec rozpoznaného textu	Rozpoznávač vrací řetězec rozpoznaného textu
Vrácení výsledku	Výsledek je zaslán skrze Websockety	Výsledek je vložen do těla HTTP odpovědi
Přijetí odpovědi	Přijme výsledek v reálném čase	Výsledek přijat v těle HTTP odpovědi

Mnoho rozpoznávačů využívá technologii HTTP požadavků. Avšak jak je vidět, HTTP dotaz/odpověď mechanismus je znevýhodněn oproti Websocketům. Chybějí mu podstatné rysy a musí zasílat vždy HTTP hlavičku k dotazu. Z toho plyne, že websockety jsou o mnoho víc použitelné, rychlejší a efektivnější na síťové rozpoznávaní hlasu oproti HTTP dotaz/odpověď mechanizmu.

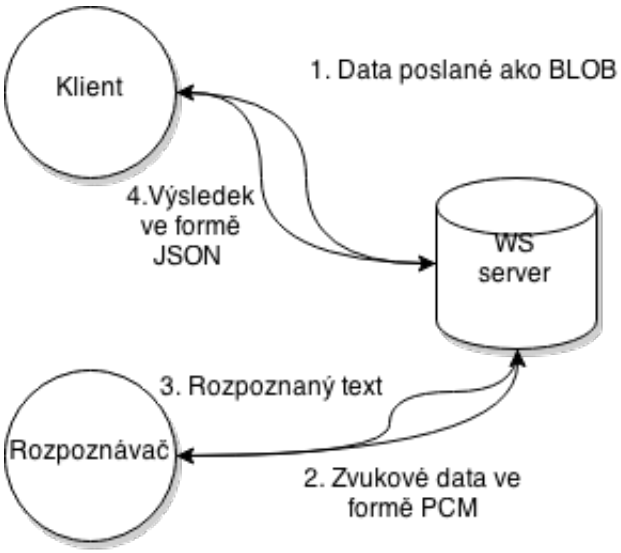
Kapitola 5

Návrh aplikace

V rámci této práce bylo navrženo a implementováno rozhraní na komunikaci klienta se serverem, Android a Web aplikacemi. Aplikace zatím slouží pro ukázkové účely, avšak rozhraní se bude moc používat pro již kompletní aplikace. Nám dodaný řečový rozpoznávač umožňuje rozpoznávaní ve více jazycích (angličtina, čeština, mandarínská čínština, ...), což umožňuje rozšířit rozhraní pro více lidí. V této kapitole bude popisovány specifikace požadavků na aplikaci a její síťový a strukturovaný návrh.

Obrázek 5.1 zobrazuje jaké budou základní části našeho systému. Bude se skládat z následujících částí:

- Klientská aplikace
- Websocketový server
- Rozpoznávač hlasu



Obrázek 5.1: Síťové rozpoznávaní řeči s použitím technologie WebSocketů

5.1 Specifikace požadavků

Specifikace byla rozdělena na více částí. První potřebnou úlohou bylo nastudovat rozhraní rozpoznávače hlasu od společnosti Google, který umožňuje online diktovaní a přepis řeči. Rozhraní je přístupné jenom pro chromium vývojáře s tím, že maximální počet requestů na server, pro účel rozpoznávaní je 25. Využívají RESP API, s tím že data jsou zasílána ve formátu JSON.

Další částí bylo nastudovat daný rozpoznávač řeči a vybrat vhodný protokol pro komunikaci. Bylo potřebné nastudovat vytvoření instance rozpoznávače offline. Rozpoznávač řeči je popsán v kapitole 3 a výběr protokolu v kapitole 4.

Třetí částí specifikace bylo implementovat rozhraní, pro daný rozpoznávač, který bude umožňovat klient-server komunikaci. Klient bude používat WebSocketový protokol jak je popsáno v kapitole 4. Další detaily o návrhu budou popsány v této a další kapitole. Hlavní požadované funkce budou:

- Rozhraní umožnující klient-server komunikaci
- Napojení na rozpoznávač
- Obsluha více klientů najednou

V úkolech specifikace je taktéž navrhnout a implementovat příkladnou demo-aplikaci na platformě Android. Hlavními požadavky na aplikaci:

- Možnost připojit na server
- Nahrávaní audia a posílaní na server v časových intervalech
- Zobrazování výsledků rozpoznávaní uživateli

Završením specifikace je vytvoření testů, pro danou aplikaci, na platformě Android.

5.2 Serverová část

V této části se zaměříme na popis implementace rozhraní na straně serveru. Bude obsahovat části:

- **Server** - bude poskytovat klient / server komunikaci skrze WebSocketový protokol. Základním úkonem bude přijímat nové připojení, zprávy od připojených klientů a korektní ukončování spojení.
- **Rozhraní mezi serverem a rozpoznávačem** - bude zprostředkovávat komunikaci mezi klientem a rozpoznávačem řeči. Bude úzce spolupracovat se serverem. Při novém klientském spojení vytvoří novou instanci serveru. Při nové zprávě se zvukovými daty pošle daná data rozpoznávači řeči pro rozpoznání.
- **Rozpoznávač** - hlavní náplní bude přijímat zvukové data a vytvořit přepis. Jak už bylo zmíneno budeme využívat rozpoznávač řeči od firmy Phonexia.

Tyto části budou zaobaleny v jedné aplikaci. Ta bude smět obsluhovat více klientů, vytvářet nové instance serveru a tak dále.

Server bude přijímat požadavky od klienta. Každý klient bude mít vlastní instanci rozpoznávače. Bude se rozhodovat na základě typu zprávy. Jestli přijme textovou zprávu, nastaví rozpoznávač dle daných parametrů. Jestli přijdou binární data, jsou předány rozpoznávači na rozpoznání.

5.3 Klientská část

Tato část popisuje jak je navržená aplikace u klienta. V našem případe je navrhnuta a vytvořena aplikace pro Web a plaftormu Android. Hlavní úlohou naší demonstrační aplikace bude možnost připojení na server, na který se bude následně odesílat nahraný zvuk v časových intervalech na server.

5.3.1 Android aplikace

Android

Softwarová platforma Android[1][4] je rozsáhlá opensource plaforma, která vznikla zejména pro mobilní zařízení, jako jsou chytré telefony, PDA, navigace, tablety a tak dále. První verze byla vydána 5. listopadu 2007 firmou Open Handset Alliance(OHA[2]), pod kterou spadá více než 30 výrobců hardwaru a softwaru pro telekomunikační společnosti. Téhož dne byla plaforma Android předána společností Google pravě této firmě. Původní nápad vznikl už v roce 2003 firmou Android Inc.. Zahrnuje v sobě operační systém založený na jádře Linux, middleware, uživatelské rozhraní a aplikace. Při vývoji jsou brány v potaz omezení, kterými disponují klasické mobilní zařízení, jako je výdrž baterie, menší výkonnost, nebo málo dostupné paměti.

Programovaní na systému Android

Programování na systém Android je proces, při kterém vzniká nová aplikace. Aplikace může být naprogramována v jazyce Java nebo pro kriticky důležité části v C/C++. Vytvořené aplikace mají formát Android package(koncovka .apk), který je rozbalitelný podobně, jako soubory s příponou zip. Jsou ukládané ve složce /data/app , ke které má přistup jenom root(s bezpečnostních důvodů). APK soubor obsahuje .dex soubory, což jsou kompilované byte kódy pro DVM).

Tvorba aplikací na platformu Android je odlišná od tvorby programů na PC. Aplikace je složena z komponent, které mezi sebou komunikují pomocí intentů a navíc může používat aktivity z jiné aplikace. Aplikace musí být přizpůsobena různým velikostem zařízení, od chytrých hodinek a malých displejů mobilních telefonů, až po tablety a televize. Zároveň musí být brána v potaz i omezení, jako je výkon zařízení nebo výdrž baterie a podobně.

SDK

Android Software Development Kit je kompletní sada vývojářských nástrojů. Patří mezi ně debugger, emulátor, knihovny, dokumentace, ukázky kódů nebo tutoriály. V současnosti je možné používat SDK na jakémkoliv Linux(moderní desktopový distribuce), Mac OS X 10.5.8+ nebo Microsoft Windows XP a vyšším.

SDK jde ruku v ruce s novými aktualizacemi, verzí systému Android, čímž umožňuje vývojářům poskytnou nejnovější funkce. Avšak taktéž podporuje starší verze Androidu, pro případ, že by se chtěl vývojář aplikací zaměřit, na nižší verzi Androidu. Některé ze základních nástrojů SDK jsou :

[1] http://www.android.com/
[2] http://www.openhandsetalliance.com/

- **Android Virtual Devices (AVD)** - je správce virtuálních zařízení. Virtuální zařízení dovoluje emulovat specifické nastavení(typ, velikost paměti, výkon, verzi Androidu a tak dále).
- **Android Debug Bridge (ADB)** - je univerzální nástroj pro příkazovou řádku, který umožňuje komunikaci s instancí emulátoru nebo mobilním telefonem se systémem Android. ADB je klient-server program a obsahuje 3 složky:
  - **Klient** - běží na vývojářském počítači. Z příkazové řádky posílá příkazy na server.
  - **Server** - běží jako proces na pozadí na vývojářském počítači a zprostředkovává komunikaci mezi klientem a ADB daemonem, bežícím na emulátoru nebo zařízení.
  - **Daemon** - je běžící proces na pozadí každého emulátoru nebo reálného zařízení.
- **Dalvik Debug Monitor Service (DDMS)** - používá se při ladění aplikací. Dovoluje sledovat a spravovat vzdálené procesy na emulovaných či reálných zařízeních, logovat chyby, pořizovat snímky obrazovky, sledovat nebo spravovat data a tak dále.

Aplikace

Základní ukázková aplikace bude implementována na platformě Android. Ve specifikaci je požadavek distribuování Android aplikace v obchodě od společnosti Google s názvem Google Play. Pro dosažení toho cíle je nutno vytvořit developerský účet, skrze který je povoleno publikovat hry do této služby.

Uživatel pro použití naší aplikace bude muset udělat určité kroky. Jako první krok je stáhnout si aplikaci z Google Play. Po stažení se aplikace sama nainstaluje. Po instalaci a spuštění se uživatel může připojit na server. Po připojení je možnost zapnout nahrávaní a začít mluvit. Aplikace po přijetí zobrazí text na displeji uživatele, kde si ho může uživatel prohlédnout.

Aplikace je rozdělena do několika modulů, které jsou na sobě nezávislé, ale spolupracují spolu. Jsou to moduly:

- **Websocketový klient** - má na starosti připojení na server. Po žádosti o připojení se pokusí vytvořit spojení serverem na příslušný koncový uzel(endpoint), dle nastavení jazyka(cs nebo en).
- **Hlasový nahrávač** - má za úlohu nahrávat hlasový záznam, který se poté bude v pravidelných intervalech zasílat na server. Audio se bude nahrávat v raw PCM rámcích.

5.3.2 Webová aplikace

Webová aplikace je v mnoha věcech stejná, liší se jednom implementační platforma a jazyk. Důležitou částí Webové aplikace je potřeba mikrofonu. V mobilních telefonech, s platformou Android, je mikrofon samozřejmostí, avšak ne každý uživatel Webové aplikace musí mít mikrofon.

Webová aplikace bude mít modul navíc, který bude zobrazovat aktuální analýzu a zobrazí zvukovou vlnu poslanou na server.

5.4 Posloupnost událostí

Při spuštění aplikace, má klient možnost připojit se na server. Po připojení na server klient může nahrávat zvuk. Ten se bude ukládat do formátu PCM Wave. Každou sekundu se bude posílat nahraný zvuk na server skrze technologii Websocketů. Při připojení klienta server zažádá o API-klíč a ověří oprávnění uživatele o připojení k serveru. Při příjmu dat je přeposílá na zvukový konvertor, který data dále zpracuje a přepošle do rozpoznávače. Rozpoznávač vrátí výsledek a přepošle ho až na server, který jej skrze WebSockety pošle klientovi. Uživateli se zobrazí text na obrazovce.

Kapitola 6

Implementace serverové části

Tato kapitola se zabývá popisem implementace serverové části a popisem využitých technik. Jak už bylo zmíněno v předešlých kapitolách, aplikace bude umožňovat klient-server komunikaci. Bude využívat technologii WebSocketů, která bude využitelná i do budoucna. Celá serverová část je implementována v jazyce C++.

6.1 Části serverové aplikace

V této nejdůležitější části, rozhraní propojuje server s rozpoznávačem řeči a server s klientem. Mezi základní moduly tedy patří:

- **WebSocketový server**
- **Rozpoznávač řeči od firmy Phonexia**
- **Rozhraní mezi rozpoznávačem a serverem**

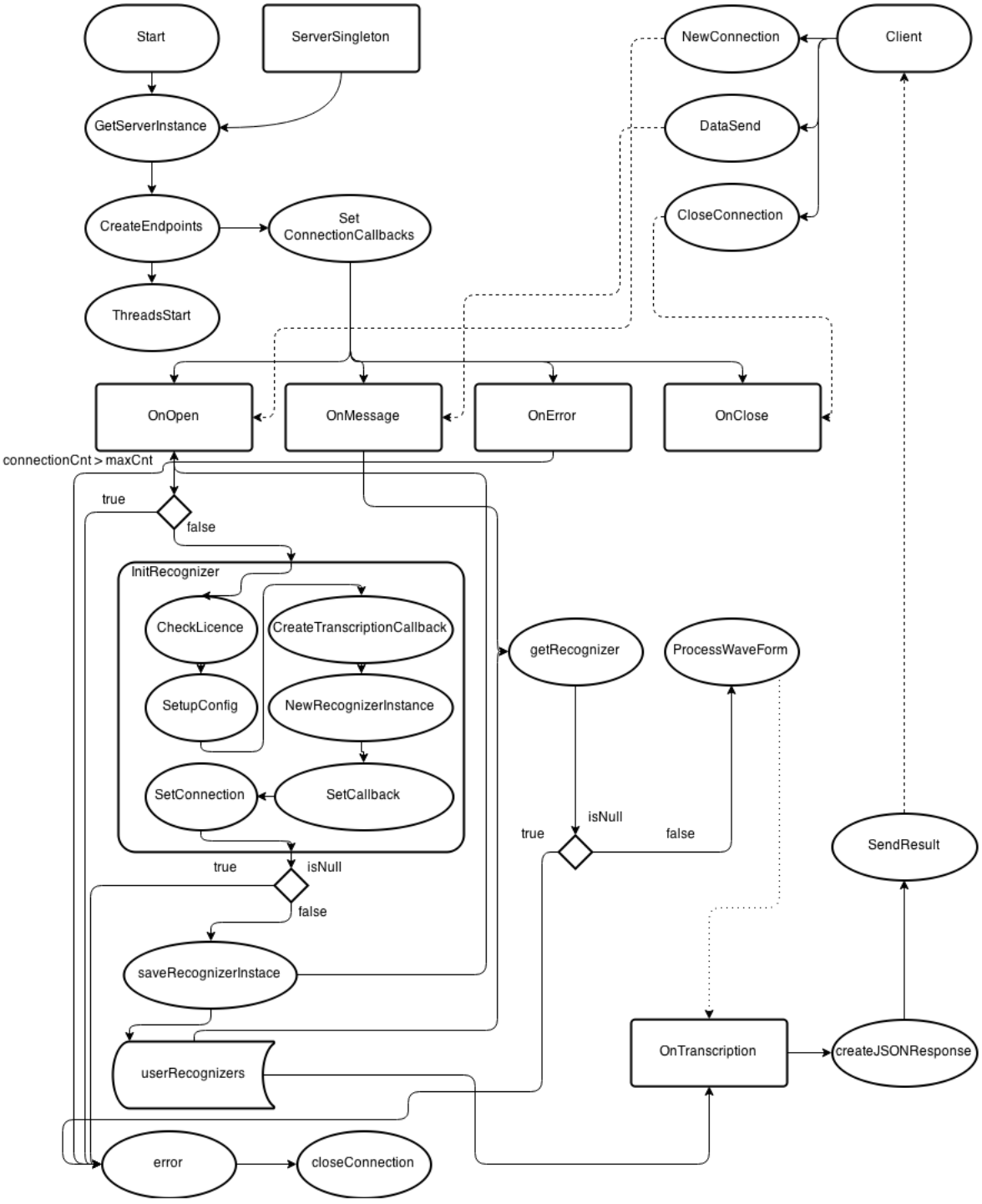
6.2 Websocketový server

Jak už bylo zmíněno, server je implementován v jazyce C++. Jeho hlavní rysy jsou:

- Z velké části podporované RFC 6455. K dispozici jsou testové/binární rámce, ping-pong, zrušení spojení se statusem a důvodem chyby a tak dále.
- Je využívaný takzvaný *Thread Pooling*. Při vytvoření serveru se vytvoří určitý počet vláken, který obsluhuje kliente. Vlákno je klientovi přiřazeno automaticky, podle vytížení vláken, čímž se rozloží práce.
- Platformově nezávislý.
- Implementovány časovače(timeouts) pro spojení s klientem. Odpojí klienta podle nastavení timeout_request a timeout_idle, když čas přesáhne nastavenou hodnotu.
- Jednoduché pro přidání nových koncových uzlů(endpoints) s použitím regexů a anonymní funkce.

Server je závislý na použití knihovny boost[1] a OpenSSL knihoven[2].

[1] http://www.boost.org
[2] https://www.opensssl.org



Obrázek 6.1: Diagram zobrazující cyklus serverové aplikace

6.3 Rozpoznávač hlasu

Jak bylo řečeno, naše řešení bude využívat rozpoznávač řeči od firmy Phonexia. Při dodaní rozpoznávače bylo třeba nastudovat, jak vytvořit instanci vlastního rozpoznávače. Ukázkovou aplikací byl **online_rec**, který ukazoval použití rozpoznávače, přímo s mikrofonem nebo s nahrávkou. Nejdůležitější částí celého systému je přibalený objektový .so soubor rozpoznávače.

Pro vytvoření jednoduchého rozpoznávače je potřebné vytvořit si instanci *SOnlineSpeechRecognizerI*. Při vytvoření se nastavuje konfigurační soubor, kterým se vybírá rozpoznávaný jazyk (nastaveno v konfigu). Dále se kontroluje licence skrze licenční server společnosti Phonexia. Pro zachycení chyb se musí nastavit *callback* [3], který se volá při chybě v rozpoznávači. Dědí rozhraní od *SErrorCallbackI*. Po vytvoření instance rozpoznávače je potřeba nastavit target callback, který je volán, pokud je něco rozpoznáno. Ten dědí z rozhraní *STransCallbackI* a uvnitř se vypisuje podle nastaveného formátu výpisu.

6.4 Rozhraní pro server

Naše rozhraní je spojení mezi rozpoznávačem a serverem. Musí umožňovat připojení klienta na server k určitému koncovému uzlu(endpoint). Pro každý rozpoznávaný jazyk je vytvořen jeden koncový uzel. Pro češtinu je to *cz* a pro angličtinu je to *en*. Klienti se potom připájí na server podle požadavku jazyka rozpoznávaní. Například, pokud se chce klient připojit na server, který má adresu 12.18.5.108 a požadovaný jakyk je angličtina, připojí se na URI "*http://12:18:5:108:8080/en*".

Instance našeho serveru je vytvořena jako Singleton. Singleton pattern zabezpečuje, že bude existovat jenom jedna instance serveru. Je implementován v třídě *ServerSingleton*. Toto řešení bylo potřebné kvůli využívaní instance v různých vláknech zároveň. Od verze C++11 nám singleton pattern zabezpečuje takzvanou *thread-safe initialisation*, která zaručuje bezpečný přístup k proměnným.

Na obrázku 6.1 jde vidět základní životní cyklus serveru. Po spuštění aplikace se inicializuje instance serveru skrze *ServerSingleton*. Poté se vytvoří koncové uzly pro dané jazyky a spustí se vlákna. Pro angličtinu uzel *en* a češtinu *cz*. Pro každý uzel se vytvoří handlery pro události. Tyto události jsou:

- **onOpen** - tato událost se volá po připojení klienta na daný uzel. Zjistí se, jestli si uživatel může vytvořit instanci rozpoznávače. Jestli ne, server ukončí spojení s klientem.

  Vytvoření instance probíhá ve funkci *initRecognizer*. Jako první se otestuje platnost licence na licenčním serveru společnosti Phonexia. Poté se nastaví config dle daného koncového uzlu(vybere se jazyk rozpoznávání). Dalším krokem je vytvoření instance třídy *STransTarget*, která zaručuje přijímaní událostí od našeho rozpoznávače. Poté se vytvoří instance rozpoznávače, nastaví se handler na události, vytvořený v předešlém kroku a nastaví se spojení, na které se budou odesílat výsledky rozpoznávání. Poté funkce vrátí instanci rozpoznávače, která se uloží mezi ostatní rozpoznávače. Ukládá se to ve tvaru klíč spojení - instance rozpoznávače.

  Důvodem, proč je potřeba sdílet instance mezi vlákny je to, že při každém požadavku reaguje na zprávu jiné vlákno aplikace(to právě volné) a potřebuje přistupovat taktéž k již vytvořené instanci.

[3] Callback je část spustitelného kódu, který se přenáší do jiné funkce skrze parametr

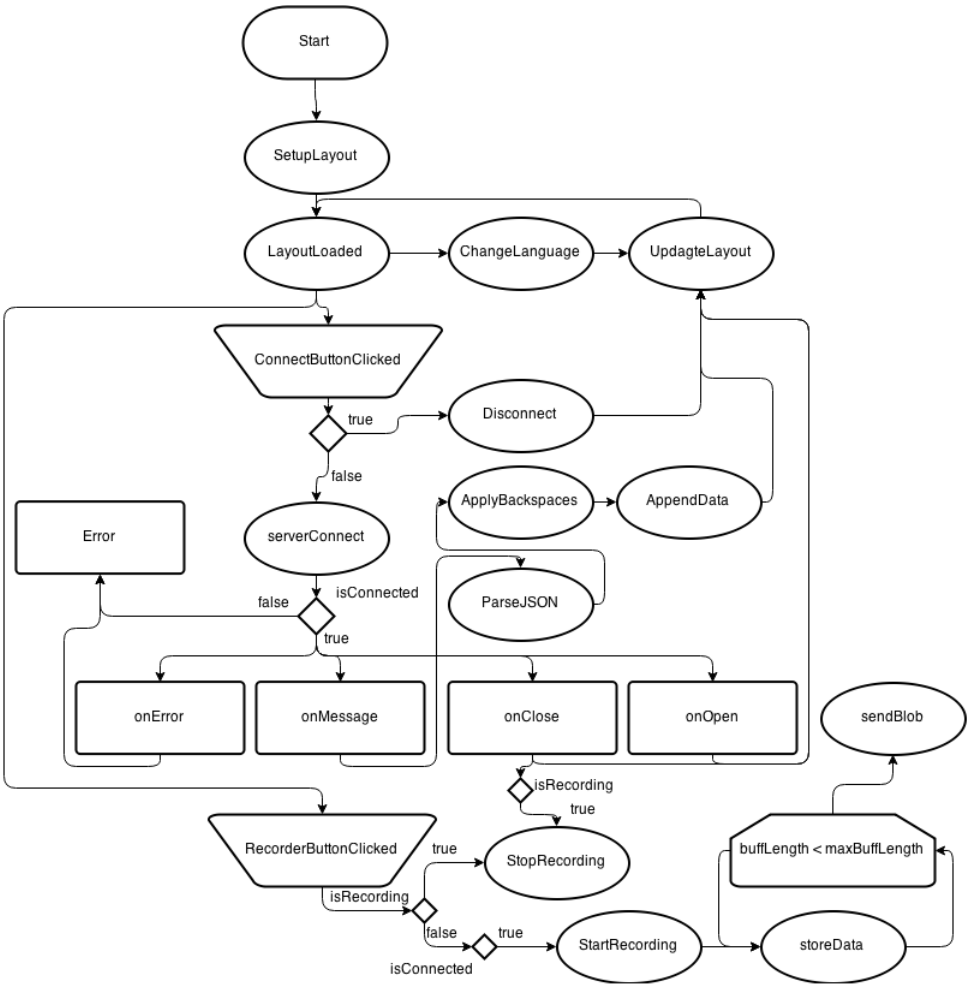
- **onClose** - při ukončení spojení se najde instance odpovídající příslušnému spojení a vymaže se z paměti.
- **onError** - při chybě se spojení ukončí a vymaže se odpovídající instance.
- **onMessage** - při zprávě se najde odpovídající instance rozpoznávače, dle klíče daného spojení. Ze zprávy se seberou data a vloží se do funkce *ProcessWaveForm*, která bere na vstupu raw PCM rámce. Po zpracování těchto dat, rozpoznávač vyvolá událost. *onTranscription*.

V události *onTranscription* se nejprve zjistí počet smazaných znaků(počet znaků typu backspace), skrze metodu *GetRemovedCharacters* v třídě *STransWordHistoryI*. Poté se z této třídy metodou *GetWord* postupně získávají slova. Ty vytvářejí řetězec, který je následovně dán do formátu JSON, s počtem smazaných znaků a poslán klientovi.

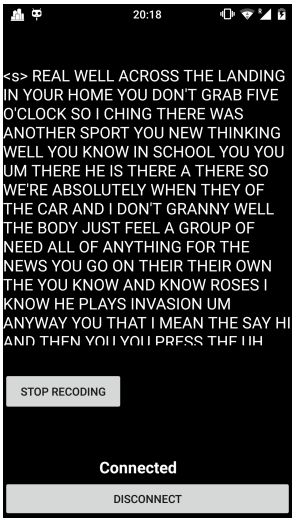
Kapitola 7

Implementace klientské části

Jak bylo zmíněno v předešlých kapitolách, v této části se zaměříme na implementaci klienta na platformy Android a Web. Obě aplikace musí splňovat základní požadavky specifikace. V části 7.1 si popíšeme implementaci aplikace na platformě Android a v části 7.2 si popíšeme implementační detaily z aplikace pro Web.



Obrázek 7.1: Jednoduchý diagram zobrazující cyklus klientské aplikace



Obrázek 7.2: Ukázka vzhledu Android aplikace

Na obrázku 7.1 jde vidět základní cyklus naší klientské aplikace. Po spuštění aplikace je potřebné vykreslit uživatelské rozhraní. Poté je aplikace připravena na použití. Uživatel si může vybrat jazyk pro připojení na server. Dostupné jazyky jsou angličtina a čeština. V aplikaci se nacházejí 2 tlačítka, která obsluhují 2 hlavní akce.

První akcí je připojení/odpojení na server. Při kliknutí na *Připojit* se klient pokusí připojit na server na konkrétní koncový uzel podle nastavení jazyka. Po úspěšném připojení na server se nastaví potřebné *event handlery*(funkce reagující na určité události) a překreslí se rozhraní. Jestli je uživatel připojen, je možné se po stisku tlačítka odpojit od serveru.

Druhou důležitou akcí je nahrávaní zvuku skrze mikrofon. Dle nastavení se po stisku tlačítka začne nahrávat zvuk(jestli se již nenahrává, jinak se zastaví nahrávání). Během nahrávaní se data ukládají do bufferu, dokud se nenaplní na dostatečnou velikost. Poté jsou odeslána na server. Data se neposílají ihned, poněvadž by se posílala v intervalech od 5-10 ms, což by zahltilo server požadavky od klienta. Velikost výstupního bufferu je nastavena tak, aby čas odesílaní zhruba odpovídal 1 sekundě.

Po přijetí zprávy od serveru (ve tvaru JSON) klientská aplikace zjistí potřebné údaje. Server zasílá JSON ve tvaru:

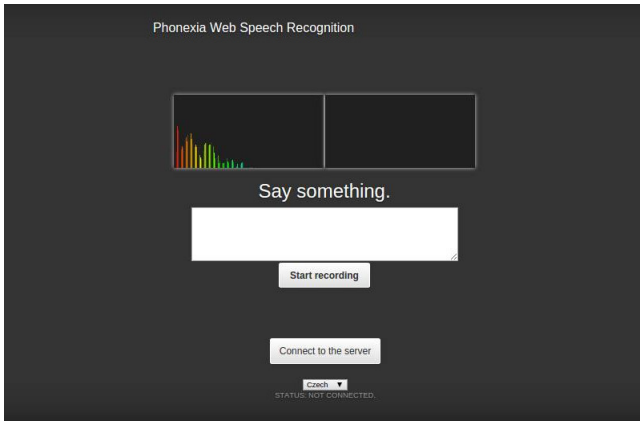
```
{
  "backspaces" : 10,
  "data" : "I love unicorns"
}
```

Z dat, která přicházejí od serveru jde zjistit počet znaků, které je třeba smazat z již rozeznaných slov a data, která třeba k nim připojit.

7.1 Aplikace pro platformu Android

V předešlé kapitole v části 5.3.1 jsme si stanovili a upřesnili hlavní části aplikace. Naše aplikace bude distribuována prostřednictvím mobilního internetového obchodu Google Play. Každý uživatel si ji bude moci stáhnout bezplatně. Pro použití aplikace bude muset každý uživatel splnit pár jednoduchých kroků:

- Uživatel si najde naši aplikace v Google Play.



Obrázek 7.3: Ukázka vzhledu aplikace

- Před stáhnutím a instalací, si aplikace vyžádá potřebné povolení pro používaní mikrofonu a síťového připojení.
- Po povolení se aplikace sama stáhne a nainstaluje, čímž je připravena pro používání.

Naše aplikace je napsána v jazyce Java pro platformu Android. Jak už bylo řečeno v části 5.3.1, Android je softwarová platforma pro mobilní telefony. Minimální verzi API, kterou bude podporovat naše aplikace, je verze 9, což odpovídá Androidu s názvem Gingerbread(2.3). V současné době, se tím zaměříme na témě ř 99.5% mobilních zařízení, se systémem Android.

Jak bylo navrhnuto, naše aplikace se skládá s vícero částí:

- **WebSocketový klient** - je založený na knihovně *JavaWebSocket*(verzie 1.3.0), kterou šlo nainstalovat pomocí systému Gradle[1]. Knihovna umožňuje připojit se na WebSocketový server, přijímat nebo zasílat textové a binární zprávy a odchytávaní chyb spojení.
- **Nahrávač řeči** - Pro nahrávání řeči byla použita základní Android třída *MediaRecorder*. Umožňuje nastavení frekvence, reprezentaci zvuku(mono,stereo), zvukový formát a tak dále.

  V naší aplikaci jsme využili frekvenci 44.1 kHz, mono zvuk a 16 bitovou PCM reprezentaci dat. Nahrávaní se spouští stiskem tlačítka *Start Recording*. Avšak je to možné jen tehdy, pokud je klient připojen na serveru. Pokud je uživatel připojen, zavolá se funkce *StartRecording*, která vytvoří nové vlákno(na UI vlákně by neměli běžet takovéto typy úloh, snižuje se tím výkonnost a uživatelská přívětivost).
- **MainActivity** - třída, která obsluhuje naši hlavní aktivitu. Její instance běží na UI vlákně. Při inicializaci se nastavují handlery, pro události tlačítek, které, poté, tato třída bude obsluhovat. Obsluhuje také zobrazení rozeznaného textu a tak dále.

7.2 Aplikace pro Webovou platformu

V kapitole 5.3.2 bylo zmíněno, že ne každý počítač má zabudovaný mikrofon. Pro naši aplikaci je mikrofon nutností, protože bez ní nejde nahrávat hlasové nahrávky. Použití

[1] https://gradle.org/

mikrofonu v prohlížečích internetu je taktéž velice obtížné. V současné době ne každý prohlížeč podporuje používaní mikrofonu a je dáno že čím vyšší verze prohlížeče, tím lepší. Aplikace je psána za použití technologie HTML5 a veškeré funkční části jsou napsány v jazyku Javascript. Na obrázku 7.3 je vidět vzhled naší aplikace.

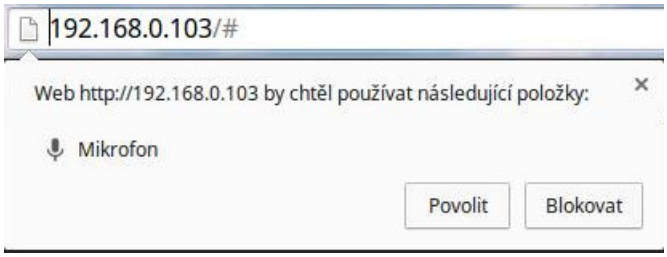
Naše aplikace se skládá z více části:

- **Spojení na server aneb Websocketový klient** - pro připojení na server, aplikace využívá vestavěnou technologii websocketů, uvnitř webového prohlížeče. Dle výběru nastaveného jazyka(implementováno jako html select element) se připojí na konkrétní koncový uzel(endpoint). Pro český jazyk to je cz a pro anglický je to en. Po připojení na server je možné okamžitě začít s nahráváním hlasu a rozpoznáváním skrze server.
- **Nahrávač řeči** - je založený na Javascriptové knihovně Recorder.js. Má možnost nastavení pro stereo nebo mono zvuk, dále má možnost exportovat nahrávku a uložit ji na lokální úložiště. Pro naše účely je zajímavý *callback* s názvem **onmessage**. Ten se volá každou sekundu, když je třeba poslat data na server. Jako parametr přijdou data, ve formátu raw PCM a jsou zaslána na server skrze WebSocketové připojení.
- **Zobrazovač analýzy řeči a zvukové vlny** - pro vykreslení se využívá element *plátno* [2] (canvas) z technologie HTML5, na které je možné kreslit. Naše aplikace na plátno vykresluje pomocí obdélníků. Všechno obsluhuje funkce drawBuffer.

Následující proces vede k úspěšnému využití rozpoznávače skrze internet:

- Klient si otevře webovou aplikaci.
- Webový prohlížeč požádá uživatele o povolení/blokování používaní mikrofonu, za účelem nahrávaní zvukových záznamů. Jde vidět na obrázku 7.4.
- Uživatel povolí používaní mikrofonu
- Klient má možnost vybrat si jazyk rozpoznávaní. Primárně je nastaven jazyk na češtinu.
- Uživatel stiskne tlačítko **Connect**, čímž se připojí na server k odpovídajícímu koncovému uzlu
- Klient může začít rozpoznávání stiskem tlačítka **Start Recording**, čímž se zapne nahrávaní a zasílaní dat na server.
- Klientská aplikace přijme výsledky rozpoznávaní a zobrazí je uživateli v patřičném elementu aplikace.

[2] http://www.w3schools.com/html/html5_canvas.asp



Obrázek 7.4: Žádost o povolení použití mikrofonu od webového prohlížeče

Kapitola 8

Testy

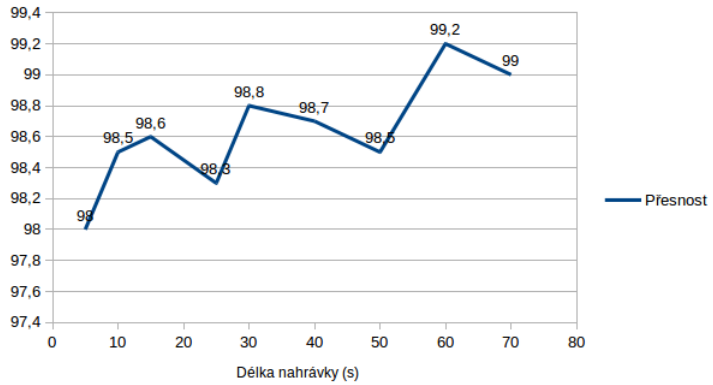
Hlavním úkolem této práce bylo implementovat rozhraní mezi serverem a rozpoznávačem hlasu, které bude umožňovat klient/server komunikaci. V této kapitole jsou popsány testy na naši implementaci. Testování bylo rozděleno na více částí:

- Porovnávání výsledků online a offline rozpoznávání řeči
- Měření časové délky rozpoznávaní online, vzhledem na délky nahrávky

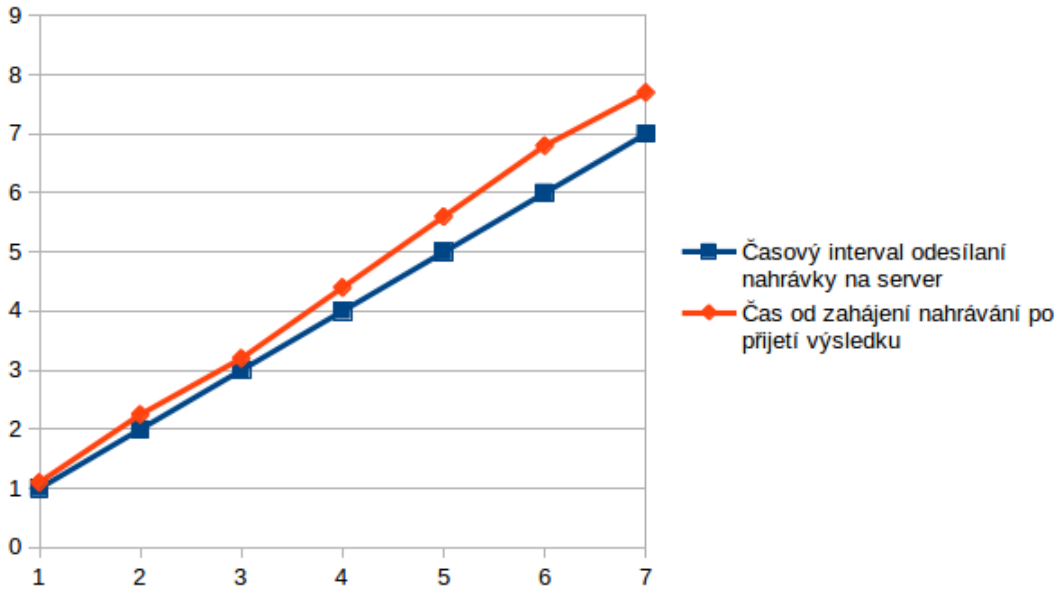
8.1 Porovnávání výsledků online a offline rozpoznávání řeči

V těchto testech jsme se zaměřili na porovnání výsledků rozpoznávaní při využití vzdáleného a lokálního rozpoznávače. Princip testu spočíval v porovnání výstupů offline rozpoznávače a online rozpoznávače, při stejné zvukové nahrávce. Pro testování bylo třeba ukládat nahrávaný zvuk u klienta a ukládat přepis. Poté se zvuková nahrávka vloží na vstup rozpoznávače a uloží se její výstup. Následovně se porovnají výsledky obou rozpoznávání. Testy byly provedeny pro klientskou aplikaci na platformě Android.

Na obrázku 8.1 je výsledek našich testů. Spodní část grafu zobrazuje délku nahrávky v sekundách. Graf zobrazuje míru podobnosti souborů, které obsahují výsledky jednotlivých rozpoznávaní. Jak jde vidět, rozpoznávaní online není 100%, což může být způsobeno špatným mazáním opravovaných slov oproti výstupu rozpoznávače.



Obrázek 8.1: Graf zobrazující míru podobnosti rozpoznávaní online, oproti offline rozpoznávaní



Obrázek 8.2: Graf zobrazující dobu nahrávky a celkovou dobu rozpoznávaní

8.2 Měření časové délky rozpoznávaní online vzhledem na délky nahrávky

Druhým testováním bylo měření času přijetí výsledku, od délky intervalu zasílaní nahrávky na server. V obou případech se měření zahájilo při zahájení nahrávání. Na obrázku 8.2 jde vidět poměr těchto dvou veličin. S grafu vyplývá, že, se stoupající dobou nahrávky, se zvedá i doba rozpoznávání. Tím pádem naše časová hodnota intervalu byla zvolena optimálně, vzhledem na "odezvu"(rozeznaný text zobrazený u uživatele) serveru a zátěž serveru.

Kapitola 9

Závěr

Cílem této práce bylo se navrhnout a implementovat rozhraní, které bude umožňovat vzdálené rozpoznávání řeči a vytvořit příkladnou demo aplikaci, která poběží na platformě Android. Naše aplikace využívá funkce rozpoznávaní, které jsou implementovány v knihovně BSAPI.

Tato práce se v prvních kapitolách zaměřila na vysvětlení problematiky rozpoznávaní řeči, kde dále následuje popis a rozhodování o výběru vhodného protokolu, pro komunikaci server s klientem. Dále je popsána architektura softwarové platformy Android, návrh aplikace, popis její implementace a nakonec testování.

Náš systém síťového rozpoznávání řeči byl implementován s použitím nových technologií Websocketů. Ty poskytují výhodu oproti HTTP protokolu a do budoucna mají lepší perspektivu využití. Systém je rozdělen do části klientské a serverové. Klientská část obsluhuje klienta, zasílá data na server a přijímá výsledky rozpoznávaní. Serverová část se stará o přijímání dat, přeposílá je do rozpoznávače hlasu a odesílá výsledek klientovi. Umožňuje vícero jazyků, pro rozpoznávaní, dle nastaveného koncového uzlu.

Tímto naše aplikace dovoluje využívat rozpoznávaní hlasu na mobilních telefonech, kde by skrz slabý výkon zařízení, baterku nebo malou paměť, nebylo možné. Rozpoznávače jsou plněny akustickými modely, které při dobrém tréninku dosahují velikosti až několik GB.

Do budoucna bychom chtěli vylepšit komunikaci se serverem a implementovat zabezpečené šifrované posílání dat tzv. wss nebo vytvořit generování takzvaných API klíčů, dle kterých by se zjišťovalo, zda má klient přistup k serveru nebo ne.

Literatura

[1] Huang, X.; Acero, A.; Hon, H.-W.: *Spoken Language Processing*. Prentice Hall, 2001, ISBN 0-13-022616-5.

[2] Institue, I. S.: TRANSMISSION CONTROL PROTOCOL. RFC 793, IEFT, September 1981.
URL https://www.ietf.org/rfc/rfc793.txt

[3] Lawrence, J.: *Introduction to Neural Networks*. California Scientific Software Press, 1994, ISBN 1-883157-00-5.

[4] Meier, R.: *Professional Android 4 Application Development*. Paperback, may 2012, ISBN 978-1-118-10227-5.

[5] Muller, M.: *Information Retrieval for Music and Motion*. Springer, 2007, ISBN 978-3-540-74047-6.

[6] Postel, J.: User Datagram Protocol. RFC 768, IEFT, august 1980.
URL https://www.ietf.org/rfc/rfc768.txt

[7] Stevens, W. R.; Fenner, B.; Rudoff, A. M.: *Unix Network Programming*. Pearson Education, Inc, 2004, ISBN 0-13-14155-1.

Příloha A

Obsah CD

- projekt.pdf - písemná zpráva
- doc/ - zdrojový tvar písemné zprávy
- src/ - zdrojové soubory
  - CMakeList.txt - soubor pro překlad aplikace
  - server.cc/server.hh - rozhraní se serverem
  - networking/ - zdrojové soubory serveru
  - recognizer/ - soubory rozhraní s rozponávačem
- html/ - zdrojové soubory webové aplikace
- android/ - zdrojové soubory Android aplikace
- poster-thesis.png - plakát prezentující naši práci
- video-thesis.mp4 - video prezentující naši práci

Příloha B

Uživatelská příručka

Naše rozhraní je implementováno v jazyce C++ a využívá nejnovější verzi C++. Pro překlad aplikace bude potřeba využít překladače s podporou C++14, například g++ verze 4.9.2. Jako další je potřeba mít nainstalovanou openssl a boost-dev knihovny. Pro překlad je používán program cmake spolu s makefile. Je potřbené nastavit proměnnou CXX(neo C++) aby odkazovala na správný překladač. Po přeložení aplikace je vygenerován soubor ws_server, kterým lze spustit server. Pro překlad a spuštení lze využít script autorun.sh, který jse nachází ve stejné zložce jako zdrojové soubory a stará se o běh serveru. Po spuštení je možné jse připojit na server na port 8080 se základně maximálne 5 nastavenými instancema. Pro změnění těchto hodnot je třeba změnit soubor server.hh.

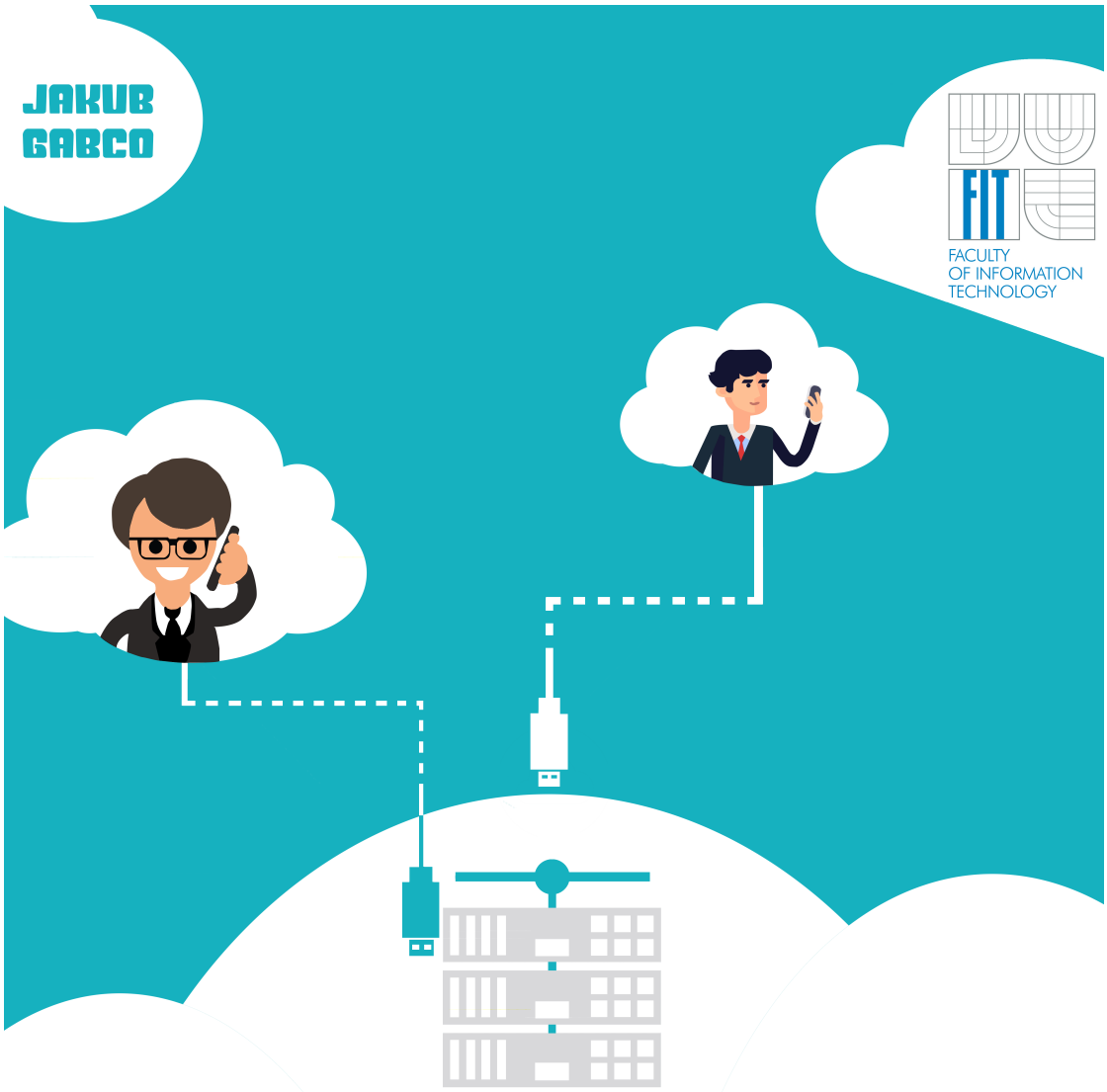
Zdrojáky, které je nutné mít od společnosti phonexia

- src/ - naše složka projektu
  - licence/ - licenční soubory
  - data/ - modely pro rozpoznávání
  - config/ - konfigurační soubory rozpoznávače
- dic/ - knihovna rozpoznávače řeči

Pro spuštení klienta je třeba změnit ip adresu:

- Android - je potřeba změnit url adresu v zdrojovém souboru MainActivity.java. Poté je třeba preložit aplikaci a niní je možné jse připojit na server.
- Web - je potřeba zmenit url adresu v zdrojovém souboru script.js. Poté stačí otevřít soubor index.html a môžete se připojit na server.

Android aplikace jse dá stáhnout i v obchodé GooglePlay, avšak spuštení ostré verze serveru je v přípravě.



SPEECH RECOGNITION ONLINE

- RAW PCM INPUT
- JSON OUTPUT
- HANDLING MULTIPLE CLIENTS SIMULTANEOUSLY
- ENGLISH AND CZECH LANGUAGE RECOGNITION
- ANDROID AND WEB PLATFORM IMPLEMENTATION
- WEBSOCKET PROTOCOL

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

ANDROID

HTML 5 WebSockets

Phonexia